ONLINE COMPUTER MAINTENANCE UTILIZING A VIRTUAL MACHINE MONITOR

BACKGROUND

[0001] Typical maintainance operations on server-class computers include replacing faulty hardware and adding new hardware (e.g., I/O adapter boards, memory cards, CPUs), upgrading and patching operating systems, upgrading and patching software applications, and performing server reconfiguration. The maintenance may be performed offline. Offline maintenance on a server can involve stopping or rebooting the server's operating system, and shutting off power to the server. These steps lead to an accumulation of "downtime" (that is, time during which the applications are not providing service). While the server is down, hardware or software or both can be serviced. After servicing, the server may be rebooted, and the applications restarted.

[0002] It is desirable for businesses to minimize server downtime. Downtime can cause interruptions in critical services, result in a loss of customers (who, inconvenienced by poor service, seek better service elsewhere), and reduce productivity of employees. In addition, the downtime can increase the cost of server operation.

[0003] Servers having online maintenance capability can avoid downtime in certain instances. The online maintenance capability can be engineered into computer hardware and operating systems. A server having online maintenance capability can allow maintenance on particular subcomponents without stopping the system as a whole. For example, a server equipped with "PCI Hot Plug" allows a PCI card to be added or replaced while applications that do not depend on that card continue to run.

Similarly, operating systems with "dynamic kernel modules" allow software components in dynamic modules to be added, replaced, or removed at runtime. Only those applications that are dependent on that module need restart.

[0004] However, not all servers have online maintenance capability. Retrofitting these online maintenance capabilities into existing servers and their operating systems and applications can be prohibitively challenging. Moreover, a redesigned legacy system having online maintenance features does not help those who want to continue using an existing version of that system.

SUMMARY

[0005] According to one aspect of the present invention, online computer maintenance is performed on at least one node. A virtual machine monitor is run; a first operating instance is run on the virtual machine monitor; a second operating system instance is run on the virtual machine monitor as a substitute for the first instance, and the maintenance is performed with respect to one of the instances while using the other of the instances.

[0006] Other aspects and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] FIG. 1 is an illustration of an exemplary node in accordance with an embodiment of the present invention

[0008] FIG. 2 is an illustration of a method in accordance with an embodiment of the present invention.

[0009] FIGS. 3a-3d are illustrations of a method of servicing hardware in accordance with an embodiment of the present invention.

[0010] FIG. 4 is an illustration of a method of adding hardware in accordance with an embodiment of the present invention.

[0011] FIG. 5a is an illustration of a method of performing operating system maintenance in accordance with an embodiment of the present invention.

[0012] FIG. 5b is an illustration of a method of performing application maintenance in accordance with an embodiment of the present invention.

[0013] FIG. 6 is an illustration of a method of performing online computer maintenance in accordance with another embodiment of the present invention.

[0014] FIG. 7 is an illustration of a method of performing online computer maintenance in accordance with yet another embodiment of the present invention.

DETAILED DESCRIPTION

[0015] As shown in the drawings for purposes of illustration, the present invention is embodied in a method for performing online computer maintenance on a node. The maintenance includes, but is not limited to servicing hardware and software, performing hardware and software reconfiguration, tuning hardware, and modifying software (e.g. upgrading, patching, tuning). Examples of servicing these resources includes adding new memory; replacing or adding I/O devices, adapters and controllers; replacing faulty CPUs in multiprocessors; upgrading, patching, reconfiguring and rebooting operating systems; and upgrading and patching applications.

[0016] An exemplary node 110 is illustrated in FIG. 1. The node 110 includes a computer 112 having a processing unit 114, memory 116, and other hardware 118. The processing unit 114 may include a single processor or multiple processors. Memory for storing a virtual machine monitor (VMM), applications, and operating system(s) may be located in the computer 112 or another node. The computer 112 is not limited to any particular type. Examples of computers include file servers, web servers, workstations, mainframes, personal computers, personal digital assistants (PDAs), print servers, and network appliances.

[0017] Reference is made to FIG. 2, which illustrates a method of performing online computer maintenance. The VMM is run on the node (210). The VMM is a layer of software that runs between a physical layer (raw hardware) of the computer and an operating system ("OS") layer. The VMM may be loaded during bootup of the computer. At boot time, the VMM can receive control of the hardware, and may maintain hardware control until the computer is shut down.

[0018] One or more OS instances are run on the VMM (212). The VMM allows multiple OS instances to run simultaneously in the computer. When the VMM starts an OS instance, it may configure the hardware to trap when the OS instance executes privileged instructions (e.g., instructions that perform I/O, instructions that load a translation lookaside buffer). When one of these traps occurs, the VMM simulates the instruction and thus creates an illusion that the OS instance has sole control of the hardware on which it runs. The OS instances may be instances of the same operating system, or instances of different operating systems.

[0019] A first one of the OS instances is used (214). For example, the OS instance may be used by running applications on it. A web server is but one example of an application.

[0020] At some point in time, a decision is made to perform online maintenance on the node. For example, the decision might be made by a system administrator.

[0021] A second operating system instance is run as a substitute for the first instance (216). The substitute (second) OS instance is intended to provide the same services as the first OS instance by taking over the services temporarily or permanently. The second OS instance is run on the same VMM. The first and second OS instances may be instances of the same or different operating systems.

[0022] Maintenance is performed with respect to one of the first and second instances while using the other of those instances (218). Thus, maintenance can be performed with respect to the first OS instance while the second OS instance is being used, or maintenance can be performed with respect to the second OS instance while the first OS instance is being used. The choice will depend upon the type of computer maintenance that is performed. This will become clear from the examples below of hardware and software maintenance.

[0023] Since at least one of the first and second OS instances is being used during the maintenance, the maintenance is performed without incurring downtime. As a benefit, applications can be run on the one OS instance while the other OS instance is being serviced.

[0024] If online maintenance requires the second OS instance to be used while maintenance is performed with respect to the first OS instance, applications can be migrated from the first OS instance to the second OS instance. This application migration can take several forms. Application migration for web servers, for example, can be as simple as redirecting requests sent to that application to an instance of the application already running on the second OS instance. In other cases, administrators can migrate an application by terminating it on the first OS instance and restarting it on the second OS instance. Data structures of the application may have to be restored on the second OS instance. Transparent process migration can also be used to move running applications from one OS instance to another. With process migration, the first OS instance or runtime environment encapsulates the process user-level and kernel-level data structures, moves them to the second OS instance, and recovers the process there in a manner that is largely or entirely transparent to the application. Techniques from failure recovery literature, such as full-process checkpointing, message logging, and process pairs can be used to implement process migration. Once the applications have been migrated, a user can continue using the applications that are running on the second OS instance.

[0025] Application migration could occur from the first OS instance to the second OS instance, and back to the first OS instance. Application migration could occur before the maintenance is performed, or after the maintenance is performed. The specifics will depend upon the type of computer maintenance that is performed. This too will become clear from the examples below of hardware and software maintenance.

[0026] Additional steps may be performed after the online maintenance has been completed (220). For example, the first OS instance may be rebooted, and the second OS instance may be shut down. Or the second OS instance may continue in place of the first OS instance. The additional steps will depend upon the type of computer maintenance that is performed, as will become clear from the examples below of hardware and software maintenance.

[0027] The steps 216-220 can be initiated by a system administrator. The system administrator can run the substitute OS on the VMM (216), perform the maintenance (218), and complete any additional steps (220). In the alternative, these steps 216-220 can be initiated by software.

[0028] The VMM is designed to support these steps 216-220. For example, the VMM might be designed to prevent or shield an OS instance from seeing the hardware to be removed or added; trapping and emulating probes of I/O space by an OS instance; etc. This list of functions is not exhaustive. These additional functions will depend upon the type of computer maintenance that is performed. The VMM could even be designed to initiate these steps 216-220.

[0029] The method is not limited to a single node. A single node was described above merely to simplify the description of the online maintenance method. In a cluster environment having multiple nodes, the online maintenance may be performed on multiple nodes simultaneously, without requiring spare nodes. Applications could not only migrate to other nodes in the cluster, but could also migrate from one OS instance to another within the same node.

[0030] Several examples of online computer maintenance will now be provided. A first example relates to hardware removal, a second example relates to hardware addition, and a third example relates to software maintenance.

[0031] Reference is now made to FIGS. 3a-3d, which illustrate an example of hardware removal. A computer is already running a VMM 312 on hardware 310, a first OS instance 314 on the VMM 312, and an application 316 on the first OS instance 314 (FIG. 3a). The first OS instance 314 has a dependency on the hardware to be removed.

[0032] The VMM 312 creates a second OS instance 318, which does not have a dependency on the hardware to be removed (FIG. 3b). When the second OS instance is created, it attempts to discover hardware in the computer. However, the VMM can prevent or shield an OS instance from seeing the hardware to be removed. Shielding can be performed in any number of ways. As an example, a platform has firmware that handles hardware resource discovery by probing the hardware and filling in a hardware description table. The firmware then passes the table to a booting OS instance to identify the hardware resources. Before, the table is passed to the booting OS instance, however, the VMM or other privileged software entity modifies the table to prevent the OS from seeing the hardware to be removed.

[0033] As another example, the VMM can trap I/O space "probes", and misinform the booting second OS instance about the result of a probe. During boot, an OS typically determines a hardware configuration by probing I/O space and by querying various registers maintained by the hardware. For example, the booting OS instance might read an address associated with each slot in a PCI bus. A value of –1, for example, could indicate that the slot is empty. By trapping those probes and queries and setting the values (e.g., returning a value of –1), the VMM 312 can make sure that the second OS instance 318 doesn't discover the hardware to be removed.

[0034] The VMM 312 migrates the application 316 from the first OS instance 314 to the second OS instance 318 (FIG. 3c), and thereafter shuts down the first OS instance (FIG. 3d). Before removing the hardware, the system administrator may invoke a program that instructs the node to prepare for device removal. For example, if the hardware change includes removing a card from a bus slot, the system administrator may invoke a program that powers down the bus slot used by that card.

[0035] Before the hardware is removed, the VMM 312 releases its own dependencies on that hardware. In many instances, this will involve shutting down the VMM driver (if any) for the hardware to be removed. The VMM 312 may also adjust its own hardware map and interrupt tables. RAM removal is slightly more complicated because the VMM's data structures can depend quite broadly on the memory to be removed. To eliminate the dependency of the VMM 312 on that memory, the VMM 312 can move any code or data it is maintaining on those physical pages to a new location (a safe range of memory), and continue to use the code or data at the new location. If the VMM 312 is designed to use virtual memory, each in-use page can be backed from the region to be moved with a new page of physical memory (by making simple changes to the page table of the VMM), and copy the old pages to the new. However, if the VMM 312 uses directly physical memory, then the code and data structures of the VMM 312 is constructed with sufficient indirection to allow their backing pages to be copied to another location.

[0036] The hardware is removed safely while the user uses the second OS instance 318 and the application 316 running on the second OS instance 318. The only time the application stopped during this maintenance was during migration. Although the length of this stoppage will depend on the particular migration technique employed, migrating an application from one OS instance to another on the same physical machine can be performed very quickly so as to be barely noticeable (if noticeable at all) to a user.

[0037] This first example can be extended to servicing other than hardware removal. For example, switches on the hardware can be flipped, pins can be set, and so forth while the second OS instance is being used. After the hardware has been serviced, the VMM could run its resource discovery algorithm to rediscover the serviced hardware.

[0038] Reference is now made to FIG. 4, which illustrates an example of adding hardware to a node. A VMM is already running in the node, a first OS instance is running on the VMM, and an application is running on the first OS instance. For example, the application is a web server, and the system administrator decides to add a memory card a month after bootup.

[0039] The administrator begins by running a program that instructs the node to prepare for hardware addition (410). For example, if the hardware addition includes adding a card to a bus slot, the program powers down that bus slot. Hardware is then added to the computer (412).

[0040] While the hardware is being added, the VMM hides the existence of the added hardware from the first OS instance (414). This may include shielding the first OS instance from interrupts by the added hardware, and trapping and emulating probes of I/O space by the first OS instance.

[0041] The VMM discovers the added hardware (416). For some servers, this may entail rerunning the VMM resource discovery algorithm. Other platforms may generate a CPU interrupt upon hardware addition, which the VMM then services. If the VMM is maintaining device drivers for the added device, the VMM triggers the relevant device driver's initialization routine. The VMM adjusts its own hardware maps and interrupt tables so that interrupts from the new hardware are serviced by the correct routines.

[0042] The VMM runs a second OS instance on the virtual machine monitor after the hardware has been added (418). The VMM allows the second OS instance to discover the added hardware (418). Exposing the hardware can involve letting I/O space probes by the booting OS "see" the existence of the added hardware. As a result, the added hardware is configured into the booting instance.

[0043] After the second OS instance has been booted, the application is migrated from the first OS instance to the second OS instance (422), and the VMM shuts down the first OS instance (424). The user continues to use the application running on the second OS instance. Thus hardware is added without incurring downtime.

[0044] Reference is now made to FIG. 5a, which illustrates an example of performing OS maintenance on a computer already running a VMM and a first OS instance. A second OS instance is run on the virtual machine monitor (510); applications are migrated from the first instance to the second instance (512).

[0045] The software maintenance can be performed before or after the applications are migrated. The software maintenance can be performed on the applications, operating system or both. One of the instances may be shut down after maintenance and application migration have been performed.

[0046] As a first example, a second instance is booted on the VMM, where the second instance is a new version of an operating system. The applications are migrated to the second instance after the second instance has booted, and the first instance is shut down after the application migration.

[0047] As a second example, a second instance is booted on the VMM, where the second instance is an old version of the operating system. After the second instance has been booted, it is patched or upgraded. The applications are migrated from the first instance to the patched second instance. The first instance is shut down after the application migration.

[0048] As a third example, the second operating system is booted, the applications are migrated to the second instance, and the first instance is reconfigured. Reconfiguration might include changing tunable parameters, changing a domain name or IP address, etc. After being reconfigured, the first instance is restarted. Applications are migrated from the second instance to the reconfigured first instance. The second instance is shut down after application migration.

[0049] Reference is now made to FIG. 5b, which illustrates an example of performing maintenance on an application. Prior to maintenance, a first instance of the application is running on a first OS instance, and the first OS instance running on a VMM. The application maintenance may include running a substitute OS instance on the VMM (550), running a second instance of the application on the substitute OS instance (552), and making changes to the second application instance (554). After the changes have been made, a cut over is made from the first application instance to the second application instance (e.g., by redirecting a request stream from the first application instance to the second application instance). The first OS instance and first application instance are then shut down (556).

[0050] The first and second application instances could be run on the same OS instance instead of different OS instances. However, there are advantages to running the first and second application instances on different OS instances. One advantage is the avoidance of conflicts between the two application instances.

[0051] The virtual machine monitor adds overhead to the node. This overhead can reduce performance of the node. However, various approaches may be used to reduce the overhead.

[0052] As a first example, the virtual machine monitor partitions hardware to reduce the overhead. Such a virtual machine monitor is disclosed in assignee's U.S. Serial No. _________ filed ____ (attorney docket no. 200208635-1), which is incorporated herein by reference.

[0053] As a second example, the first OS instance is booted on the hardware prior to running the VMM (610). When the maintenance is to be performed (612), the VMM is interposed beneath the first operating system instance, the second OS instance is run on the VMM (614), and the maintenance is performed with respect to one of the OS instances while using the other of the instances (616). This method is illustrated in FIG. 6. Interposition of a VMM is disclosed in assignee's U.S. Serial No. _________ (attorney docket no. 200208633-1), which is incorporated herein by reference.

[0054] A third example is illustrated in FIG. 7. A VMM is run on a node (710), and maintenance is performed (712). After the maintenance has been performed, the VMM is devirtualized (714). The devirtualization may be partial or full. Full devirtualization includes devirtualizing the CPU, memory and I/O of the node. Partial devirtualization includes devirtualizing any one or two of the CPU, memory and I/O Devirtualization of memory is disclosed in assignee's U.S. Serial No. _________ (attorney docket no. 200300561-1), which is incorporated herein by reference; and devirtualization of I/O is disclosed in assignee's U.S. Serial No. _________ (attorney docket no.

200309154-1), which is also incorporated herein by reference. Full devirtualization is also disclosed in assignee's U.S. Serial No. ____(attorney docket no. 200208633-1). If the VMM is fully devirtualized, it may be unloaded from memory (716).

[0055] Thus disclosed are non-invasive methods of performing maintenance on a computer, without shutting down the computer. The maintenance is non-invasive because it does not require re-engineering of the operating system and hardware. The maintenance can be performed on legacy systems that don't support hot swapping and other forms of on-line maintenance.

[0056] The present invention is not limited to the specific embodiments described and illustrated above. Instead, the present invention is construed according to the claims that follow.